ROS-Cyton Module

1.Package Summary

The ROS-Cyton Module provides a ROS interface for Energid's actinSE and Robai's Cyton 7-DOF humanoid manipulators. The aim of this module is to expose actinSE and the Cyton hardware API through ROS .

2.Overview

Using the *ROS-actionlib* a user can access the methods of Actin-SE and the Cyton hardware API. The methods included allow for direct, real-time control of the Cyton robot arms in both jointspace and end-effector modes.

The module consist of six nodes named as follows: *actinSE_node* ,*cyton* ,*guide frame* ,*hardware* ,*send_joints* ,and *set home* .The core node(action server) is called *actinse*. The *actinse* node accepts end effector (EE) coordinates and publishes joint values and joint rates. The node which sends EE coordinates is called the *guide frame* node. It can be any program which is configured to send EE coordinates to a particular topic .The *cyton* node will subscribe to the output of the *actinse* node and send the valued to the cyton arm using the cyton hardware API (or the *hardware* node). The *set home* node moves the cyton to a default home position .

Cyton can also be directly controlled using joint space values. There are two nodes for doing this task, the *hardware* node and *send_joints* node. The hardware node is an action server which will receive the joint values from the *send_joint*s action client and publishes to a topic. The *cyton* node will subscribe to the topic and move the cyton according to the joint values.

3.Detailed Description

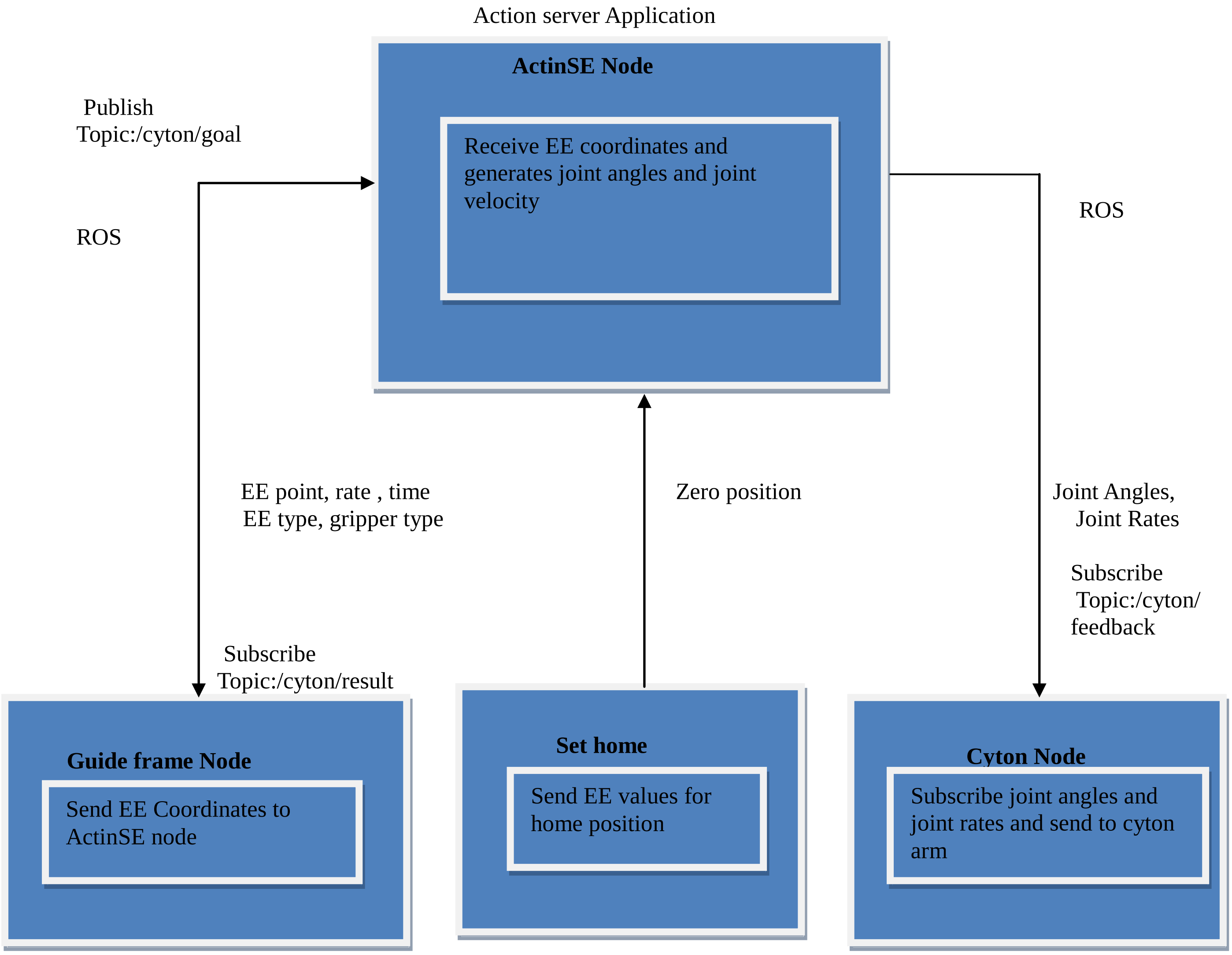
For additional details on the control interface or Actin please see the Actin-SE API or the Cyton Hardware Interface API.

3.3 ROS-Cyton Module Architecture

The ROS-Cyton module is using actinSE and the cyton-hardware library for its operation. The module mainly consist of *ActionServers* and *ActionClients*. There are two *ActionServers* and two *ActionClienst*. The first *ActionServer* is called *actinse*. It performs EE to joint-space conversion using the ActinSE Cyton IK engine. Its *ActionClient* is the *guide frame* node. It will send guide frame values to the *ActionServer*. The second *ActionServer* is the *hardware* node. It is responsible for marshalling joint values from the *ActionClient* to the /cyton/feedback topic. Its *ActionClient* is the *send joints* node .It will send the joint values to the *hardware* node .The user can control the cyton in two ways.

- End effector control
- Joint-space control

3.1 End effector Control


Action server Application
ActinSE Node
Receive EE coordinates and generates joint angles and joint velocity
Publish
Topic:/cyton/goal
ROS
ROS
EE point, rate , time
EE type, gripper type
Zero position
Joint Angles,
Joint Rates
Subscribe
Topic:/cyton/feedback
Subscribe
Topic:/cyton/result
Guide frame Node
Send EE Coordinates to ActinSE node
Set home
Send EE values for home position
Cyton Node
Subscribe joint angles and joint rates and send to cyton arm

During end effector control ,the *actinse* node acts as action server and the *guide frame* node is the action client. The EE values are published from the *guide frame* node to the *actinse* node . The *actinse* node subscribes to the values and feeds back the resulting values in the **/cyton/feedback** topic**.**

3.1.1 actinSE node

This node acts as the *ActionServer*. The node will subscribe to topics publish by the action client.

Subscribed Topics

- **/cyton/goal**

Consist of the following messages:

float32[] position :EndEffector coordinates or jointValues
float32[] rate :Joint-rates

float32 time :Simulation time
int32 eeindex :EndEffector type
uint32 home :Home flag to move cyton to home position
float32 gripper_value :Gripper joint angle for controlling gripper separately
float32 gripper_rate :Gripper joint rate

Published Topics

- **/cyton/result**

  Consist of the following messages:

  float32[] position :Joint values

- **/cyton/feedback**

  Consist of the following messages:

  float32[] position :Joint Values
  float32[] rate :Joint rates
  float32 time :Simulation time
  float32 gripper_feed_value
  float32 gripper_feed_rate

3.1.2 guide_frame node(input node)

This node act as the action client. It will publish the following topics:

Published Topics:

- **/cyton/goal**

  Consists of the following messages:

  float32[] position :EndEffector coordinates
  float32[] rate :Joint-rates
  float32 time :Simulation time
  int32 eeindex :EndEffector type
  uint32 home :Home flag to move the cyton to home position
  float32 gripper_value :Gripper joint angle for controlling gripper separately
  float32 gripper_rate :Gripper joint rate

3.1.3 cyton_node

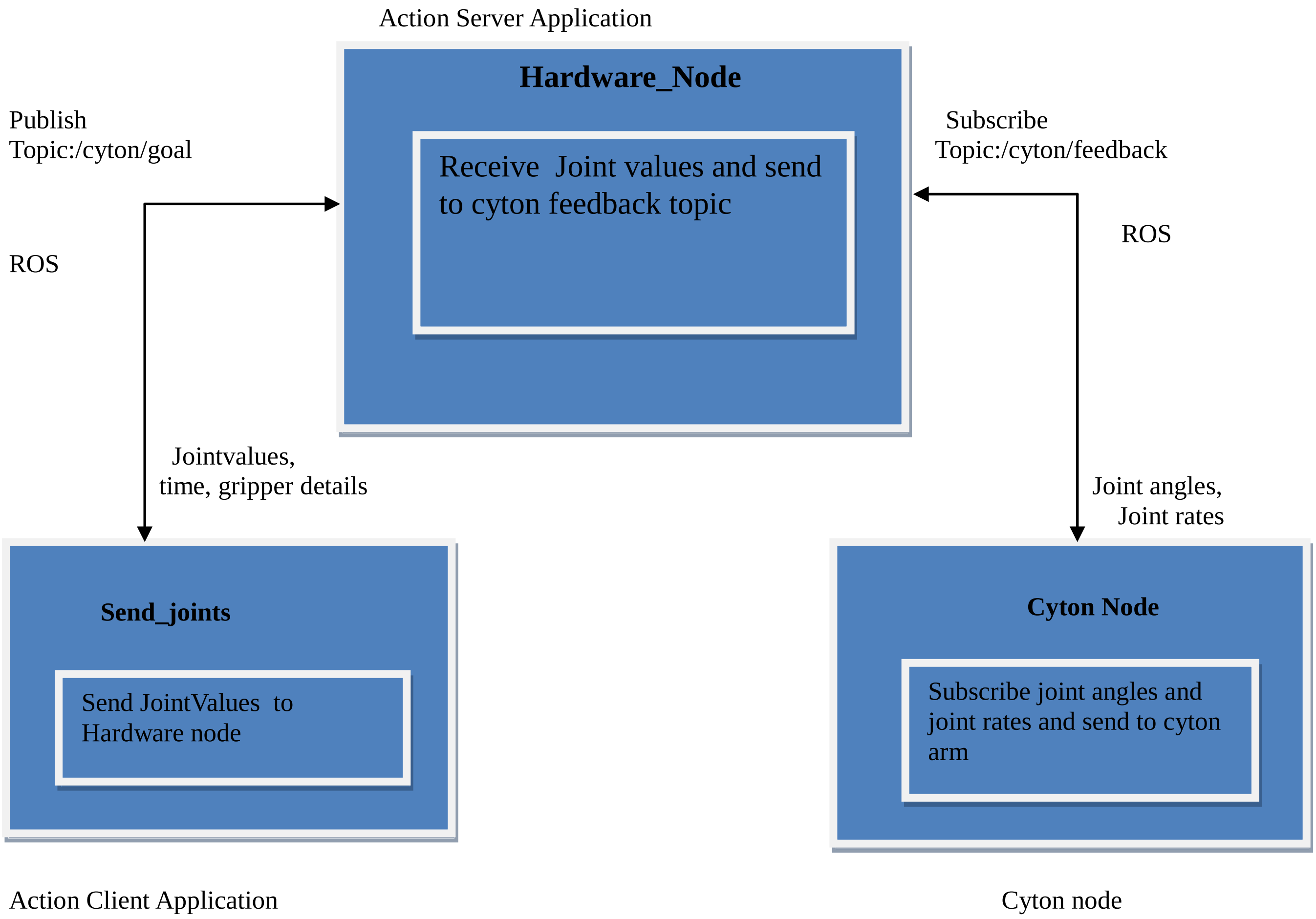
This node handles the movement of the cyton robotic arm. It uses the cyton hardware API for movement. It will subscribe to the **/cyton/feedback** topic from the *actinSE* node and push data to the hardware.

3.1.4 set_home

Moves the cyton to a default home position. It will only work when the user is working with EE coordinates .

Execution : **rosrun cyton set_home**

3.2 Joint-level Controlling



Action Server Application
Hardware_Node
Receive Joint values and send to cyton feedback topic
Publish
Topic:/cyton/goal
ROS
Subscribe
Topic:/cyton/feedback
ROS
Jointvalues,
time, gripper details
Joint angles,
Joint rates
Send_joints
Send JointValues to Hardware node
Cyton Node
Subscribe joint angles and joint rates and send to cyton arm
Action Client Application
Cyton node

3.2.1 Hardware_node

This node handles the movement of cyton through direct joint values .It acts as an *ActionServer*. It will publish joint values to the topic **/cyton/feedback** .

Subscribed Topics

- **/cyton/goal**

Consist following messages

float32[] position : jointvalues
float32[] rate :Joint-rates
float32 time :Simulation time
int32 eeindex :EndEffector type
uint32 home :Home flag to move cyton to home position

float32 gripper_value :Gripper joint angle for controlling gripper separately
float32 gripper_rate :Gripper joint rate

Published Topics

- **/cyton/feedback**

Consist of following messages

float32[] position :Joint Values
float32[] rate :Joint rates
float32 time :Simulation time
float32 gripper_feed_value
float32 gripper_feed_rate

3.2.2 send_joints

This node acts as an *ActionClient* of the hardware_node. This will publish joint values to topic **/cyton/goal.**

Published Topics:

- **/cyton/goal**

Consists of the following messages:

float32[] position :EndEffector coordinates
float32[] rate :Joint-rates
float32 time :Simulation time
int32 eeindex :EndEffector type
uint32 home :Home flag to move cyton to home position
float32 gripper_value :Gripper joint angle for controlling gripper separately
float32 gripper_rate :Gripper joint rate

3.2.3 cyton_node

This node handles the movement of the cyton robotic arm. It uses the cyton hardware API for movement. It will subscribe to the **/cyton/feedback** topic from the actinSE node and send to hardware .

- **/cyton/feedback**

Consist of following messages

float32[] position :Joint Values
float32[] rate :Joint rates

float32 time :Simulation time
float32 gripper_feed_value
float32 gripper_feed_rate

Here the action-client is called the *guide frame* node. The function of the *guide frame* node is to send guide frame values (End effector coordinates) to the actinSE node .The guide frame node can be any program which can send the guide frame coordinates .

4. Tutorials

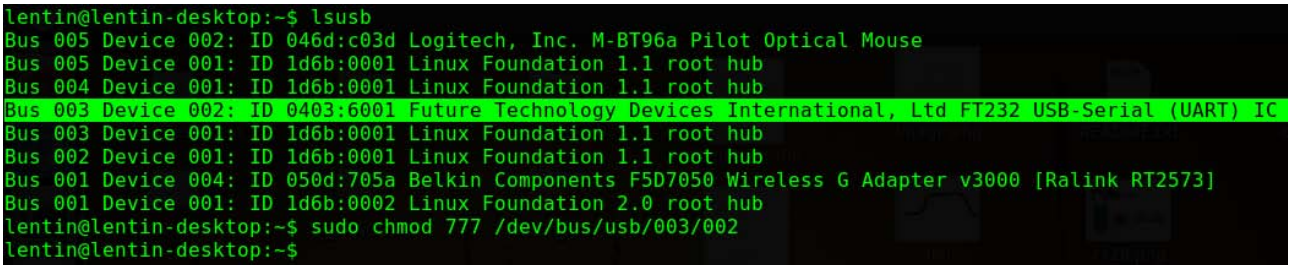
This package will works well in Ubuntu 11.04 ,for other distribution it is still in testing

4.1 Installation

1. Set the PATH instruction in README file inside the cyton module folder
2. Execute roscd cyton
3. rosmake –rosdep-install cyton

4.2Execution

Before execution user need to change the permission of FTDI device as shown in fig



```
lentin@lentin-desktop:~$ lsusb
Bus 005 Device 002: ID 046d:c03d Logitech, Inc. M-BT96a Pilot Optical Mouse
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 002: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART) IC
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 004: ID 050d:705a Belkin Components F5D7050 Wireless G Adapter v3000 [Ralink RT2573]
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
lentin@lentin-desktop:~$ sudo chmod 777 /dev/bus/usb/003/002
lentin@lentin-desktop:~$
```

For EE control

1. **roslaunch cyton cyton.launch :Initialises cyton hardware node and actinSE actionclient**
2. **rosrun cyton guide_frame_node :Send EE values to cyton**
3. **rosrun cyton set_home :Setting cyton to home position**

For Joint Control

1. **roslaunch cyton hardware.launch :Initialises cyton hardware**
2. **rosrun cyton send_joints :Send direct joint command to cyton**

README

ROS-Cyton Installation Procedure on Ubuntu
----

1)Install ROS-electric from

http://www.ros.org/wiki/electric/Installation/Ubuntu

2)Copy cyton folder(ROS-Cyton interface) to user location
For example copy cyton folder to home folder of Ubuntu

3)Add Environmental Variables such as

CYTON_INC :=include directory path inside the cyton folder
CYTON_LIB :=lib folder path inside the cyton folder
CYTON_BIN :=bin folder path inside the cyton folder
CYTON_EE_FILE :=EE file path,which stores a series of EE position .

For testing purpose there are some default values .The user can take any file name as EE file .It has to follow certain format like guide_frame.txt given as example inside the bin folder .

These variable has to add on the bottom of .bashrc on the home folder

Note:All path must be system path not relative

An example configuration is shown below

```
export CYTON_INC="/home/lentin/myworks/cyton/include/"
export CYTON_LIB="/home/lentin/myworks/cyton/lib/"
export CYTON_BIN="/home/lentin/myworks/cyton/bin/"
export CYTON_EE_FILE="/home/lentin/myworks/cyton/bin/guide_frame.txt"

#Setting ROS_PACKAGE_PATH here

source /opt/ros/electric/setup.bash
export ROS_PACKAGE_PATH=$ROS_PACKAGE_PATH:/home/lentin/myworks/cyton
```

4)Add **ROS_PACKAGE_PATH** as shown above.

5)Clear or Delete the existing Makefile and replace with the following line

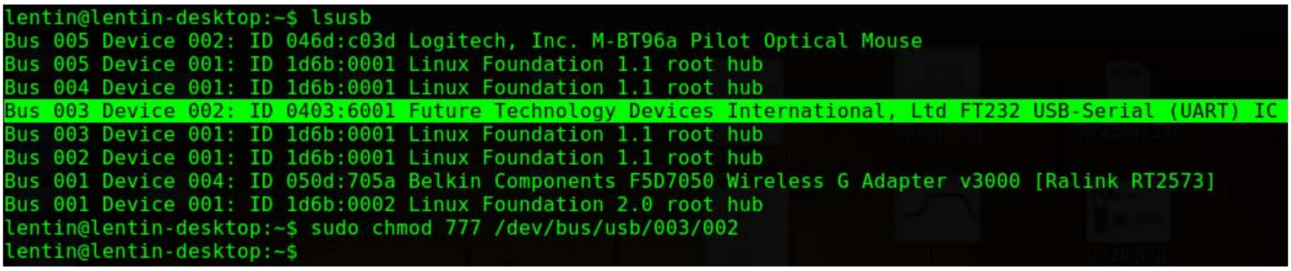
```
include $(shell rospack find mk)/cmake.mk
```

5)Open a terminal inside cyton folder and build using rosmake command

6)Install module using rosmake --rosdep-install command

Execution

Change the permission of FTDI device as shown in fig



```
lentin@lentin-desktop:~$ lsusb
Bus 005 Device 002: ID 046d:c03d Logitech, Inc. M-BT96a Pilot Optical Mouse
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 002: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART) IC
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 004: ID 050d:705a Belkin Components F5D7050 Wireless G Adapter v3000 [Ralink RT2573]
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
lentin@lentin-desktop:~$ sudo chmod 777 /dev/bus/usb/003/002
lentin@lentin-desktop:~$
```

For EE control

----

roslaunch cyton cyton.launch

rosrun cyton guide_frame_node

For setting to home position

----

rosrun cyton set_home

For Joint Control

----

roslaunch cyton hardware.launch

rosrun cyton send_joints